Attorney Docket No. 30126-8013.US01

Express Mail Label No.: EV 336 060 605 US

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF:	
Pujare *et al.*	EXAMINER: Kyung H. Shin
APPLICATION NO.: 09/826,607	ART UNIT: 2143
FILED: APRIL 5, 2001	CONFIRMATION NO: 4038
FOR: **CONVENTIONALLY CODED APPLICATION CONVERSION SYSTEM FOR STREAMED DELIVERY AND EXECUTION**	

APPELLANT'S BRIEF ON APPEAL

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is an appeal to the Board of Patent Appeals and Interferences from the decision of Examiner Shin mailed June 19, 2006, in which pending claims 1-52 stand in final rejection.

The present paper is Appellant's Appeal Brief submitted in compliance with 37 C.F.R. §41.37(c).

REAL PARTY IN INTEREST

The real party in interest is Endeavors Technology, Inc., a division of Tadpole Technology plc.

RELATED APPEALS AND INTERFERENCES

Appellants are not aware of other appeals or interferences which would directly affect or be directly affected by or have a bearing on the Board's decision in the present appeal. Stream Theory, Inc., a division of Tadpole Technology plc, is currently litigating U.S. Patent No. 6,453,334 (Case No.: SACV06-489 CJC), which may or may not be related to the Board's decision in the pending appeal. No decision has been rendered.

STATUS OF CLAIMS

The application was initially filed with 39 claims. Claims 40-52 were added by amendment on December 21, 2004. The final rejection of pending claims 1-52, as presented in Appendix A, is appealed.

STATUS OF AMENDMENTS

Appellant's response dated September 19, 2006, and submitted after final rejection was entered and made of record. All amendments submitted to date have been considered and entered by the Examiner.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 describes "a process for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment" (pg. 44, line 15 to pg. 69, line 36). The process includes "providing installation monitoring means [FIG. 4, 403] for

monitoring an installation process [FIG. 4, 402] of said conventionally coded application program [FIG. 4, 401] on a local computer system; wherein said installation monitoring means gathers modification information including system registry modifications [FIG. 23, 2317] that said installation process makes to certain file paths in a system registry of said local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The process includes "parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer [FIG. 2, 215, 216]" (pg. 18, lines 20-24; pg. 19, lines 6-9; pg. 47, lines 5-9; pg. 47, line 35 to pg. 48, line 10; pg. 49, lines 31-37). The process includes "providing data set creation means [FIG. 4, 404] for processing said modification information [from FIG. 4, 403] for converting said application program [FIG. 4, 401] into a data set [FIG. 4, 405] suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program" (pg. 17, lines 26-35; pg. 18, line 13 to pg. 19, line 15).

Independent claim 14 describes "an apparatus for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment" (pg. 44, line 15 to pg. 69, line 36). The apparatus includes "an installation monitoring module that monitors the installation process of said conventionally coded application program on a local computer system; wherein said installation monitoring module gathers modification information including system registry modifications that said installation process makes to certain file paths in the system registry of said local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The apparatus includes "a module that parameterizes said system registry modifications by replacing certain of said file paths with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer" (pg. 18, lines 20-24; pg. 19, lines 6-9; pg. 47, lines 5-9; pg. 47, line 35 to pg. 48, line 10; pg. 49, lines 31-37). The process includes

"data set creation module that processes said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program" (pg. 17, lines 26-35; pg. 18, line 13 to pg. 19, line 15).

Independent claim 27 describes "a program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment" (pg. 44, line 15 to pg. 69, line 36). The instructions include "monitoring the installation process of said conventionally coded application program on a local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The instructions include "gathering modification information including system registry modifications that said installation process makes to certain file paths in the system registry of said local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The instructions include "parameterizing said system registry modifications by replacing certain of said file paths with parameters that are recognizable by said client" (pg. 18, lines 20-24; pg. 19, lines 6-9; pg. 47, lines 5-9; pg. 47, line 35 to pg. 48, line 10; pg. 49, lines 31-37). The instructions include "processing said modification information for converting said application program into a form for deceiving said client into allowing streaming of bits of said converted application program over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program" (pg. 17, lines 26-35; pg. 18, line 13 to pg. 19, line 15).

Independent claim 40 describes "a method for converting an application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment" (pg. 44, line 15 to pg. 69, line 36). The method includes

"monitoring an installation process of an application program on a local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The method includes "gathering modification information including information on modifications that said installation process makes to certain file paths in a system registry of said local computer system" (pg. 20, lines 14-29; pg. 23, lines 19-31; pg. 44, line 15). The method includes "parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer" (pg. 18, lines 20-24; pg. 19, lines 6-9; pg. 47, lines 5-9; pg. 47, line 35 to pg. 48, line 10; pg. 49, lines 31-37). The method includes "processing said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program" (pg. 17, lines 26-35; pg. 18, line 13 to pg. 19, line 15).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds for review on appeal are:

1. Whether claims 1-6, 8-19, 21-32, 34-45, and 47-52 are unpatentable over U.S. Patent No. 6,574,618 (Eylon) in view of U.S. Patent No. 6,343,287 (Kumar) and further in view of U.S. Patent No. 6,374,402 (Schmeidler) under 35 U.S.C. §103(a).

2. Whether claims 7, 20, 33, and 46 are unpatentable over Eylon-Kumar-Schmeidler and further in view of U.S. Patent No. 6,457,076 (Cheng) under 35 U.S.C. §103(a).

ARGUMENT

I. First Ground: Obviousness (Eylon-Kumar-Schmeidler)

The first ground for review is whether claims 1-6, 8-19, 21-32, 34-45, and 47-52 are unpatentable over U.S. Patent No. 6,574,618 (Eylon) in view of U.S. Patent No. 6,343,287 (Kumar) and further in view of U.S. Patent No. 6,374,402 (Schmeidler) under 35 U.S.C. §103(a). A Final Office Action dated June 16, 2006 ("the Office Action") included the Examiner's rationale for finally rejecting the claims. An Advisory Action dated October 23, 2006 ("the Advisory Action"), reiterated the Examiner's rationale for rejecting the claims.

Patent examiners carry the responsibility of making sure that the standard of patentability enunciated by the Supreme Court and by the Congress is applied in each and every case. The Supreme Court in *Graham v. John Deere*, 383 U.S. 1, 148 USPQ 459 (1966), stated:

> Under §103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background, the obviousness or nonobviousness of the subject matter is determined.

A. Scope and Content of the Prior Art

1. The Scope of the Prior Art (Eylon)

"Before answering *Graham's* 'content' inquiry, it must be known whether a patent or publication is in the prior art under 35 U.S.C. §102." *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1568, 1 USPQ2d 1593, 1597 (Fed. Cir.), cert denied, 481 U.S. 1052 (1987). A 35 U.S.C. 103 rejection is based on 35 U.S.C. 102(a), 102(b), 102(e), etc. depending on the type of prior art reference used and its publication or issue date.

35 U.S.C. 102(a) states a person shall be entitled to a patent unless "the invention was known or used by others in this country, or patented or described in a

printed publication in this or a foreign country, before the invention thereof by the applicant for patent." Eylon does not qualify as 102(a) prior art because Eylon does not establish knowledge or use of the invention prior to the effective date of the present application, which is November 6, 2000, and the Eylon patent was not published until issuance on June 3, 2003.

35 U.S.C. 102(b) states a person shall be entitled to a patent unless "the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States." Eylon is not 102(b) prior art because the publication and issue dates of the Eylon patent are after the filing date of the present application.

35 U.S.C. 102(e) states a person shall be entitled to a patent unless "the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for the purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language." Eylon was not published, so 102(e)(1) is not applicable. Eylon was granted a patent on June 3, 2003, (there is no evidence of an international patent filing), so 102(e)(2) is not applicable.

2. The Scope of the Prior Art (Kumar)

"In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of the applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." *In re Oetiker*, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). Kumar is

not within the field of endeavor of streaming software. Accordingly, the Examiner must be relying upon Kumar as "reasonably pertinent to the particular problem with which the inventor was concerned."

The Examiner relies upon Kumar at page 7 of the Office Action to teach "parameterizing said registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client (see Kumar col. 1, lines 57-61; col. 1, lines 17-20: application configuration information; col. 16, lines 22-28; col. 16, lines 31-34; col. 21, lines 36-38: parameterized configuration data) to re-direct requests for reading said system registry to a registry spoofer..." The cited text, in the order cited, is as follows:

> As used herein, the term "profile information" refers to any information or meta-data used by a particular piece of hardware, software, or operating system to configure, initialize, shutdown and aide in making runtime processing decisions.
>
> These systems include application code and data that are distributed among a variety of data structures, data processor systems, storage devices and physical locations.
>
> A parameterized value serves as a placeholder for a value supplied at runtime by the profile service itself. ExternalDataReference object 501 includes an interface that accepts parameters corresponding to the parameterized values. A parameter value is swapped into the value of another attribute within the profile at the time of evaluation.
>
> Step 901 uses parameter data passed to the external data reference object at runtime to fill any parameterized values specified in the field or query data items.

Apparently, the Examiner is using Kumar to show that data exists that is associated with application code and parameterized values can be used at runtime. The Examiner's point is not particularly clear. However, the Examiner has never associated any of the cited text with a stream-enabled application. Nor has the Examiner explained the relevance of these citations with respect to parameterizing registry modifications. Therefore, the Examiner has failed to show that Kumar is reasonably pertinent to the particular problem with which the inventor was concerned.

3. The Scope of the Prior Art (Conclusion)

Since the Examiner relies upon Eylon, Kumar, and Schmeidler to reject the claims, eliminating one or more of the references as prior art would overcome the Examiner's rejections. The applicants believe Eylon is not prior art because the filing date of Eylon is later than the effective date of the present application. The applicants believe Kumar is not prior art because it is not in the field of endeavor of the applicants' inventions, and is not reasonably pertinent to the particular problem with which the inventor was concerned. For either of these reasons, the rejections are traversed.

B. Differences between the Prior Art and the Claims at Issue

1. The Prior Art (Eylon)

Eylon discloses a method and system for executing streamed applications from a server on a client system (see Abstract of Eylon). Eylon also discloses that the application to be executed is stored as a set of blocks or streamlets on a server (see col. 3, lines 57-59; Fig. 1).

In the Background, Eylon discloses two installed (e.g., conventionally coded) application embodiments. The first is a remote execution embodiment Eylon describes at col. 1, line 52 – col. 2, line 16. Notably, "this type of implementation ***requires the supported software to be installed*** on the server." (col. 2, lines 5-9, emphasis added). The second is a local delivery embodiment Eylon describes at col. 2, lines 17-52. Notably, "***[a]fter installation***, the application is executed." (col. 2, lines 20-21, emphasis added).

Eylon then discloses, in the Summary of the Invention, "The application ***does not need to be installed*** on the Client PC. Instead, the application is streamed to the client's system in streamlets or blocks which are stored in a persistent client-side cache and the system is configured such that the application can begin to execute on the client machine after only a small fraction of the application is loaded." (col. 3, lines 49-55, emphasis added). The applicants respectfully assert that the only reasonable

interpretation of Eylon's disclosure is that the prior art included applications that had to be installed prior to execution, and the stream-enabled applications of Eylon do not have to be installed before execution.

As shown in Eylon, FIG. 1, a server 12 includes a stream-enabled application for provisioning to a client 14 via a plurality of streamlets. FIG. 3 is a block diagram showing the high-level architecture of the client 14 and the server 12. As shown in FIG. 3, the server includes an application block library 171 and the client includes a streaming application 100. The application block library 171 includes streamlets that can be sent to the client as the streaming application 100 is executed. It should be noted that block is a synonym of streamlet (see, e.g., Eylon, col. 3, lines 57-59).

Notably, Eylon does not disclose the concept of converting a conventionally coded computer application program (e.g., an application that has to be installed before execution) into a data set for streamed delivery over a network.

2. The Prior Art (Kumar)

Kumar has nothing to do with streaming software. There is no text or figure that refers to streaming software, or the equivalent. Since Kumar has nothing to do with streaming software, it follows that Kumar does not disclose the concept of converting a conventionally coded computer application program (e.g., an application that has to be installed before execution) into a data set for streamed delivery over a network.

3. The Prior Art (Schmeidler)

Schmeidler discloses a system for secure delivery of on-demand content over broadband access networks includes a client application executing on a user's local computer system. (See Schmeidler, Abstract). "The client process utilizes an installation abstraction which enables a title to be executed on the local computer system ***without ever being installed***." (Schmeidler, Abstract, emphasis added).

Schmeidler explains in the Background of the Invention that "on-demand delivery of content including software and multimedia titles to date has been the requirement to have the title loaded onto the subscriber's local computer system in order to execute the title" (col. 2, lines 6-9). In other words, titles have to be ***installed*** before they can be executed.

To accomplish the stated goal of executing software without a local install, Schmeidler disclose, at col. 2, line 56 – col. 3, line 6: "A title is formatted into an electronic package that contains the title's files in a compressed and encrypted form, referred to hereafter as a briq. The briq is actually a portable, self-contained file system, containing all of the files necessary to run a particular title. Briqs are stored on a network file server, referred to hereafter as a RAFT server, accessible across a broadband network. The SCDP client treats the briq like a local file system on the user's PC. When running a title, the operating system, e.g., Windows, makes read requests to this local file system. The SCDP client, which, in the illustrative embodiment, includes a Windows Virtual Device Driver (VxD), services these requests by retrieving the requested blocks of briq data from the RAFT server. After retrieving the requested block of data, the VxD decompresses and decrypts the briq data, and passes the data onto the operating system on the user's PC."

Notably, Schmeidler does not disclose the concept of converting a conventionally coded computer application program (e.g., an application that has to be installed before execution) into a data set for streamed delivery over a network.

4. The Prior Art Distinguished

a. Kumar

Since Kumar has nothing to do with streaming software, none of the elements of any of the claims are disclosed in Kumar. Accordingly, an in-depth discussion of how the claims are distinguishable over Kumar is omitted. The Examiner's assertions with respect to Kumar are addressed later in the rebuttal to the Examiner's remarks.

b. The Independent Claims (1, 14, 27)

The preamble of claim 1 includes the language: "A process for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment." Thus, the process is directed to the conversion of a conventionally coded application into a streaming application. As noted, Eylon and Schmeidler include no disclosure regarding the conversion of a conventionally coded application into a stream-enabled application.

Claim 1 includes the language:

> providing installation monitoring means for monitoring an installation process of said conventionally coded application program on a local computer system;
>
> wherein said installation monitoring means gathers modification information including system registry modifications that said installation process makes to certain file paths in a system registry of said local computer system.

It should be noted that since Eylon and Schmeidler do not disclose installation of a conventionally coded application (other than in the Background), it follows that Eylon and Schmeidler do not disclose providing installation monitoring means for monitoring an installation of a conventionally coded application, as in claim 1. Moreover, the Background does not disclose installation monitoring means.

Claim 1 includes the language, "parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer." Since Eylon and Schmeidler do not disclose the installation monitoring means, it follows that Eylon and Schmeidler do not disclose parameterizing system registry modifications. Moreover, if the stream-enabled application's execution on a client were analogized to the installation of a conventionally coded application (as the Examiner has done), then creating parameters that are

recognizable by the client to re-direct requests for reading the system registry to a registry spoofer makes no sense. Specifically, the client would have to know how to re-direct requests prior to execution, so the application would have to be installed first, then executed. This directly contradicts Eylon's and Schmeidler's assertions that the stream-enabled application is executed without loading the entire application.

Claim 1 includes the language, "providing data set creation means for processing said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program." As previously indicated, Eylon and Schmeidler do not disclose converting a conventionally coded application program into a stream-enabled application.

For the above-identified reasons, Eylon and Schmeidler, whether considered alone or in combination, fail to disclose ***any*** of the elements of claim 1. Eylon and Schmeidler, whether considered alone or in combination, fail to disclose ***any*** of the elements of claims 14 and 27 for similar reasons.

c. Independent Claim 40

The preamble of claim 40 includes the language: "A method for converting an application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment." Thus, the method is directed to the conversion of a conventionally coded application into a streaming application. As noted, Eylon includes no disclosure regarding the conversion of a conventionally coded application into a stream-enabled application.

Claim 40 includes the language, "monitoring an installation process of an application program on a local computer system." Since Eylon and Schmeidler do not disclose installation of a conventionally coded application (other than in the

Background), it follows that Eylon and Schmeidler do not disclose monitoring an installation process, as in claim 40. Moreover, the Background does not disclose installation monitoring means.

Claim 40 includes the language, "gathering modification information including information on modifications that said installation process makes to certain file paths in a system registry of said local computer system." Since Eylon and Schmeidler do not discloses monitoring an installation process, it follows that Eylon and Schmeidler do not disclose gathering modification information as in claim 40.

Claim 40 includes the language, "parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer." Since Eylon and Schmeidler do not disclose monitoring an installation process, it follows that Eylon and Schmeidler do not disclose parameterizing system registry modifications.

Claim 40 includes the language, "processing said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program." As previously indicated, Eylon and Schmeidler do not disclose converting a conventionally coded application program into a stream-enabled application.

For the above-identified reasons, Eylon and Schmeidler, whether considered alone or in combination, fail to disclose ***any*** of the elements of claim 40.

d. The Dependent Claims (2-13, 15-26, 28-39, 41-52)

Claim 2 includes the language, "said data set creation means creates a runtime data set, said runtime data set consists of all regular application files and directories containing information about said regular application files." Eylon and Schmeidler fail to disclose a data set creation means.

Claim 3 includes the language, "said data set creation means creates an initialization data set that is the first set of data streamed from said server to said client, said initialization data set prepares said client for streaming of said runtime data set." Eylon and Schmeidler fail to disclose a data set creation means.

Claim 5 includes the language, "said data set creation means creates a versioning table that contains a list of root file numbers and version numbers for tracking application patches and upgrades, and wherein each entry in said versioning table corresponds to one patch level of an application with a corresponding new root directory." Eylon and Schmeidler fail to disclose a data set creation means or a versioning table.

Claim 6 includes the language, "said versioning table is sent to said client by said server, said client compares said versioning table with said client's root file number for the particular application program to find the necessary files required for a software upgrade or patch." Eylon and Schmeidler fail to disclose a versioning table.

Claim 7 includes the language, "providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said modification information." Eylon and Schmeidler fail to disclose an installation process, so if follows that Eylon and Schmeidler fail to disclose providing a user interface for use during the installation process, or a user interface that facilitates editing the modification information.

Claim 8 includes the language, "said installation monitoring means monitors said application program as it runs and is being configured for a particular working environment on said local computer system and records common configurations of said application program thereby allowing said common configurations to be automatically duplicated on other client machines." Eylon and Schmeidler fail to disclose an installation monitoring means, so it follows that the installation monitoring means does not record common configurations of the application program.

Claim 13 includes the language, "said installation monitoring means records a state of said local computer system before said installation process begins to give a more accurate picture of any modifications that are observed by said installation monitoring means." Eylon and Schmeidler fail to disclose installation monitoring means, so it follows that Eylon and Schmeidler fail to disclose installation monitoring means recording state.

For the above-identified reasons, Eylon and Schmeidler fail to disclose some or all of the elements of claims 2-13, which depend from claim 1. Claims 15-26, 28-39, and 41-52, which depend respectively from claims 14, 27, and 40, include language similar to that of claims 2-13. Accordingly, Eylon and Schmeidler fail to disclose some or all of the elements of claims 15-26, 28-39, and 41-52.

e. *Prima Facie* Obviousness

To establish a *prima facie* case of obviousness... the prior art reference (or references when combined) must teach or suggest all of the claim limitations. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). As has been described above, Eylon, Kumar, and Schmeidler, whether considered alone or in combination, fail to teach any of the elements of claim 1. Accordingly, unless Eylon-Kumar-Schmeidler provide a suggestion, claim 1 is allowable over Eylon-Kumar-Schmeidler. Indeed, if Eylon-Kumar-Schmeidler failed to teach, without a suggestion, even one element (or a portion of an element) of claim 1, claim 1 would still be allowable over Eylon-Kumar-

Schmeidler because the prior art reference(s) must teach or suggest all of the claim limitations.

Since Eylon-Kumar-Schmeidler do not explicitly teach the claim language, it must be determined whether the Eylon-Kumar-Schmeidler implicitly teach the claim language. However, the Examiner has provided no suggestion that Eylon-Kumar-Schmeidler implicitly teach one or more claim limitations. Instead, the Examiner insists that the streamed applications are "conventionally coded", despite the express statements of Eylon and Schmeidler, and despite the distinction drawn between conventionally coded applications and stream-enabled applications in the present application. Eylon and Schmeidler simply would not work for their intended purpose if they attempted to stream conventionally coded applications or to install stream-enabled applications—contradictory contentions that the Examiner appears to espouse.

Moreover, the Examiner insists that Eylon and Schmeidler include installation monitoring means, despite the fact that Eylon and Schmeidler clearly teach away from installation of stream-enabled applications on a client, as was described above. Eylon and Schmeidler simply would not monitor installation of even a portion of a streamed application for the purpose of converting the streamed application to a streamed application. It makes no sense.

Since the Examiner has not provided a teaching or suggestion of all of the elements (or portions thereof) of claim 1, claim 1 is allowable over Eylon-Kumar-Schmeidler, whether considered alone or in combination. Independent claims 14, 27, and 40 are allowable for similar reasons. Dependent claims 2-13, 15-26, 28-39, and 40-52 are allowable at least for depending from an allowable base claim, and potentially for other reasons, as well.

C. **Level of Ordinary Skill in the Pertinent Art**

It is not clear from the Examiner's rejections what the Examiner considers to be the level of ordinary skill in the pertinent art. The applicants respectfully assert that at a minimum, ordinary skill must include knowledge of streaming software techniques. Kumar, for example, has provided no indication that Kumar knows of streaming software techniques. Kumar's filing date is prior to the publication date of any streaming software patents of which the applicant's are aware. Kumar's cited publications are directed to LDAP protocols, not streaming software. Accordingly, the applicants believe Kumar does not have ordinary skill in the pertinent art. Moreover, one of ordinary skill in the art would not be motivated to consult Kumar for the purpose of improving streaming software techniques.

Rebuttal of the Examiner's Advisory Action Remarks

The Examiner's substantive remarks in the Advisory Action are rebutted in the order the Examiner made them. Remark 1.6 is discussed later in the Second Ground for Rejection section.

1.1 Applicant's invention discloses the processing of a conventionally coded which is defined as an application without the requirement for recompiling or recoding. (see Specification Paragraph [0072]) The Eylon (6,574,618) prior art discloses the processing of a conventionally coded application without the requirement for recompilation or recoding of the application. Eylon discloses converting the application into a mode suitable for delivery via a streaming delivery mechanism without the requirement for recompilation or recoding of the application. (See Eylon col. 3, lines 45-50: server; col. 3, lines 52-56; col. 4, lines 51-56: streamed application; col. 8, lines 49-53: monitor and management, streamed application installation on local system)

The Examiner relies upon paragraph [0072] of the applicant's specification, which is:

> [0072] The invention is embodied in a conventionally coded application conversion system for streamed delivery and execution in a computer environment. A system according to the invention converts a conventionally coded application program into a streamed application suitable for concurrent execution on a client while being streamed from a server. In addition, the invention provides a system that does not require the conventionally coded application program to be recompiled or recoded.

Contrary to the Examiner's assertion, the applicants do not define a conventionally coded application as an application that does not have the requirement of recompiling or recoding. Taken out of context, this definition has little or no meaning. Paragraph [0072] clearly juxtaposes "a conventionally coded application" on the one hand and "a streamed application suitable for concurrent execution on a client while being streamed from a server" on the other. Thus, a more appropriate use of the term "conventionally coded" is as an antonym to "stream-enabled."

The Examiner's assertion that "Eylon discloses converting the [conventionally coded] application into a mode suitable for delivery via a streaming delivery mechanism without the requirement for recompilation or recoding of the application," is without support in Eylon. The Examiner cites various portions of col. 3, lines 45-56, as support for the assertion. However, in the cited paragraph, Eylon describes a stream-enabled application stored at a server that is streamed to a client in streamlets or blocks such that 1) the application does not need to be installed on the client and 2) the application can begin to execute on the client machine after only a small fraction of the application is loaded. If this be the case, then the Examiner must be asserting that the stream-enabled application is a conventionally coded application, that the stream-enabled/conventionally coded application is installed on the server (claim 1 includes the language "providing installation monitoring means for monitoring an installation process of said conventionally coded application program on a local computer system", but Eylon states the stream-enabled application is not installed on the client), that the installation is monitored on the server, and that the stream-enabled/conventionally coded application is converted to a stream-enabled application thereby. ***The***

Examiner's assertion, with respect to claim 1, necessarily means a stream-enabled application is converted into a stream-enabled application. This makes no sense.

The Examiner asserts in the Advisory Action:

> By definition, conventionally is defined as "... Conforming to established practice or accepted standards; traditional: ...", or a standard method for the coding of a application. (http://www.answers.com/conventionally&r=67)

The Examiner's use of a dictionary definition is not called for in this case, but even if it were accepted, the oldest streaming software patent has a filing date of June 16, 1998 (U.S. Pat. No. 6,453,334 to Vinson et al.). Eylon has a filing date of December 28, 2000. At the time Eylon was filed, nobody would have referred to streaming software applications as "conventionally coded" because the technology was so new. A conventionally coded application at the time the application was filed would not run on a machine unless it was installed.

The Examiner's assertion on page 3 of the Office Action that the applications of Eylon are conventionally coded because "no mention is disclosed that the applications are not conventionally coded applications" is not supported by any cited prior art. Accordingly, the Examiner is relying on personal knowledge to support the finding of what is known in the art, and the Examiner has not provided an affidavit or declaration setting forth specific factual statements and explanation to support the finding, as required by CFR 1.104(d)(2), and as requested by the applicants.

The Examiner further states at page 3 of the Office Action that "no mention is disclosed that there is any recompilation or reconfiguration of application(s) to prepare them for streamed delivery." However, the applicants claim a process that reconfigures a conventionally coded application to a stream-enabled application. There should not be any requirement of recompilation or reconfiguration after the conversion. That Eylon

does not recompile or reconfigure the application prior to streaming simply follows from the fact that the application is already stream-enabled.

The Examiner asserts at page 3 of the Office Action that "Eylon discloses the capability to format an application for streamed delivery, which is analogous to the applicant's invention. (see Eylon col. 5, lines 53-64)." The cited text refers to dividing application files into streamlets. Notably, the application is already stream-enabled.

The Examiner asserts at page 3 of the Office Action that "registry configuration parameters must be setup and installed (i.e., some form of installation) on a client system in order to execute even a streamed application. Eylon discloses execution of a streamed application on a client system. In addition, Eylon discloses the capability to monitor application installation and processing. (see Eylon col. 8, lines 49-53)." However, Eylon discloses at col. 8, lines 49-53 monitoring a streamed application. It makes no sense that the monitoring yield a data set suitable for deceiving the client into allowing streaming of bits of data over a network to the client such that the application program is capable of beginning execution on the client prior to downloading all of the application program, as in the independent claims. If the Examiner's assertions are taken seriously, the Examiner has created a system wherein an application is streamed to a client, the data necessary for streaming is gathered at the client, and from that data the application is converted into a stream-enabled application. Again, this makes no sense.

1.2 The Eylon prior art discloses that the "... application does not need to be installed on the Client PC ...". The statement merely states that the application does not need to be installed. The Eylon prior art does not discourage installation of the application, therefore it does not teach away from application installation on a client system.

Contrary to the Examiner's assertion that a streamed application is installed on a client, Eylon states: "The application does not need to be installed on the client PC.

Instead, the application is streamed to the client's system in streamlets or blocks which are stored in a persistent client-side cache and the system is configured such that the application can begin to execute on the client machine after only a small fraction of the application is loaded." (col. 3, lines 50-56). Notably, Eylon does not provide installation as an option for any given streamed application; if the application is streamed, it is not installed.

The Examiner's position is that streamed applications may or may not be installed. However, the applicants and the prior art are in agreement in that streamed applications ***are not installed***. That is one of the advantages of streaming software: Instead of installing an application on a system, the application is streamed in chunks (Eylon refers to the chunks as "streamlets" or blocks). In this way, the streamed application can be executed on a client without an install.

1.3 The Eylon prior art discloses an application transferred from a server to a client, and the application initiates execution before the entire application has been transferred. (see Eylon col. 3, lines 52-56: initiate execution after fraction of application loaded (i.e., before entire application downloaded))

Notably, an application that executes before the entire application is downloaded is not installed. Thus, the Examiner's assertion here (1.3) directly contradicts the Examiner's previous assertion (1.2).

1.4 Applicant has argued that the reference prior art does not disclose "... redirecting registry information thereby creating a registry spoof capability ...". The referenced prior art does disclose this limitation.

Eylon discloses the capability to process an application transfer utilizing a streamed delivery mechanism. (see Eylon col. 3, lines 52-56; col. 4, lines 51-56) Eylon and Schmeidler (6,374,402) combination discloses the capability to redirect (i.e., spoof, deceive) registry information during the installation processing. By definition, "to spoof"

simulates a communications protocol (i.e. update registry information concerning application installation) by a program that is interjected into a normal sequence of processes (i.e. to client, spoof appears as a normal installation of application and is transparent to client) for the purpose of adding some useful function. (see Schmeidler col. 4, lines 43-46; col. 4, lines 54-59; col. 11, lines 44-46: manipulation of registry information (i.e., redirect, spoof) during installation process)

The Examiner takes the position that "update registry information concerning application installation" is applicable. However, Eylon does not install the application and Eylon does not disclose "registry modifications," which is the language of claim 1. Again, a problem with the Examiner's assertion is that the Examiner fails to recognize that the claims are directed to a conventionally coded application that is installed on a system and converted to a stream-enabled application.

1.5 Applicant has argued that the referenced prior art does not disclose "... parameterizing the system registry modifications ...".

Eylon discloses the streamed delivery of an application (i.e. conventionally coded application) between network-connected systems. (see Eylon col. 5, lines 45-50) Eylon and Schmeidler combination discloses the concept of registry information containing configuration data for an application. (see Schmeidler col. 4, lines 43-46; col. 4, lines 54-59; col. 11, lines 44-46) Eylon, Schmeidler, and Kumar (6,343,287) combination discloses the capability to parameterized system registry configuration information including modifications and the streaming of parameterized configuration data between system. (see Kumar col. 1, lines 57-61; col. 1, lines 17-20: application configuration information; col. 16, lines 22-28; col. 16, lines 31-34; col. 21, lines 36-38: parameterized configuration data)

The language from claim 1 to which the Examiner refers is, "parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct

requests for reading said system registry to a registry spoofer." The "system registry modifications" are determined by monitoring the installation of the conventionally coded application. Thus, file paths in system registry modifications are replaced with parameters that are recognizable at a client (later, after the stream-enabled application has been generated from the conventionally coded application). Such a parameterization would not occur at the client, as the Examiner suggests.

The appropriateness of the Examiner's assertion that Eylon's stream-enabled applications are "conventionally coded" is addressed above. The Examiner has identified portions of Schmeidler that describe execution of a stream-enabled application at a client. Specifically, Schmeidler disclose at col. 4, lines 43-46, "redirecting selected of the intercepted requests to the registry entries stored on the local computer system." Notably, although the Examiner has identified that Eylon and Schmeidler use registries, which include registry information, the Examiner has provided no indication that Eylon or Schmeidler use registry modifications. The registry information is provided in a static format that does not change and is not modified. The Examiner has identified portions of Kumar that describe parameters, but, again, there is no discussion related to registry modifications, or the parameterization thereof.

II. Second Ground: Obviousness (Eylon-Kumar-Schmeidler-Cheng)

The second ground for review is whether claims 7, 20, 33, and 46 are unpatentable over Eylon-Kumar-Schmeidler and further in view of U.S. Patent No. 6,457,076 (Cheng) under 35 U.S.C. §103(a).

A. Scope and Content of the Prior Art

1. The Scope of the Prior Art (Cheng)

"In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of the applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." *In re Oetiker*, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). Cheng is

not within the field of endeavor of streaming software. Accordingly, the Examiner must be relying upon Cheng as "reasonably pertinent to the particular problem with which the inventor was concerned."

The Examiner relies upon Cheng at page 12 of the Office Action to teach "providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said modification information. (see Cheng col. 9, lines 32-42: where GUI to examine installation data)." The cited text (col. 9, lines 32-43) is as follows:

> A sample user interface 600 for the recovery function is illustrated in FIG. 6. The user interface 600 includes a field 601 indicating the previous update to be removed as selected by the user, along with an information window 603 describing the software update. The user confirms the removal of the software update by selecting the undo button 605, or may cancel with cancel button 607. The recovery function deletes the files installed for the software update, and using the archived information created by the install monitor during the installation of the product, restores the client computer system 101 to its configuration immediately before the installation of the product.

Apparently, the Examiner is using Cheng to show that installation can be monitored. However, the Examiner has never associated any of the cited text with a stream-enabled application. Nor has the Examiner explained the relevance of these citations with respect to editing modification information. Specifically, the "changes" in Cheng refer only to whether a file has been installed. The interface of Cheng provides no access to modification information, only to a list of installed files. Therefore, Cheng does not provide an interface that facilitates editing modification information as in claim 7. Since Cheng has nothing to do with streaming software, or with editing the modification information, the Examiner has failed to show that Cheng is reasonably pertinent to the particular problem with which the inventor was concerned.

2. The Scope of the Prior Art (Conclusion)

Since the Examiner relies upon Eylon, Kumar, Schmeidler, and Cheng to reject the claims, eliminating one or more of the references as prior art would overcome the Examiner's rejections. The applicants believe Eylon is not prior art because the filing date of Eylon is later than the effective date of the present application. The applicants believe Kumar is not prior art because it is not in the field of endeavor of the applicants' inventions, and is not reasonably pertinent to the particular problem with which the inventor was concerned. The applicants believe Cheng is not prior art because it is not in the field of endeavor of the applicants' inventions, and is not reasonably pertinent to the particular problem with which the inventor was concerned. For any of these reasons, the rejections are traversed.

B. Differences between the Prior Art and the Claims at Issue

1. The Prior Art

As explained above, Eylon-Kumar-Schmeidler do not disclose installation monitoring, or any claimed elements that flow from installation monitoring. The Examiner asserts at page 11 of the Office Action that "Eylon discloses a user interface for monitor and management application installation. (see Eylon col. 8, lines 49-53: application manager, monitor and management of installation process)." Contrary to the Examiner's assertion, the cited text has nothing to do with installation monitoring. Rather, Eylon describes at col. 8, lines 49-53 an application that monitors and meters usage of streaming applications.

Cheng discloses a system and method for updating client computers with software updates. Cheng has nothing to do with streaming software. The Examiner asserts on page 12 of the Office Action that Cheng discloses "providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said modification information. (see Cheng col. 9, lines 32-42: where GUI to examine installation data)." Cheng, at col. 9, lines 32-42, describes a user interface that facilitates the removal of previously installed files.

Notably, the "changes" in Cheng refer only to whether a file has been installed. The interface of Cheng provides no access to modification information, only to a list of installed files. Therefore, contrary to the Examiner's assertion, Cheng does not provide an interface that facilitates editing modification information.

2. The Prior Art Distinguished

Claim 7 includes the language, "providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said modification information." None of the cited prior art references disclose this language. Accordingly, claim 7 is allowable over the cited prior art. Claims 20, 33, and 46 are allowable for similar reasons.

C. Level of Ordinary Skill in the Pertinent Art

It is not clear from the Examiner's rejections what the Examiner considers to be the level of ordinary skill in the pertinent art. The applicants respectfully assert that at a minimum, ordinary skill must include knowledge of streaming software techniques. Cheng, for example, has provided no indication that Cheng knows of streaming software techniques. Cheng's filing date is prior to the publication date of any streaming software patents of which the applicant's are aware. Cheng's cited publications have nothing to do with streaming software. Accordingly, the applicants believe Cheng does not have ordinary skill in the pertinent art. Moreover, one of ordinary skill in the art would not be motivated to consult Cheng for the purpose of improving streaming software techniques.

Rebuttal of the Examiner's Advisory Action Remarks

The Examiner's only substantive remark in the Advisory Action that relates to the second ground is 1.6:

1.6 Applicant has argued that the referenced prior art does not disclose "... providing a user interface that allows an operator to examine all changes made to said local computer system ...". The reference prior art does disclose this limitation....

This remark has been rebutted above in the discussion of the differences between the prior art and the claims at issue.

III. Conclusion

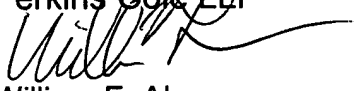
Eylon and Schmeidler are directed to streamed applications that are ***streamed rather than installed***. Claims 1-52, on the other hand, are directed to a method or system for converting a conventionally coded computer application program into a data set suitable for streamed delivery. The conventionally coded application of the claims is ***installed***.

Eylon, Kumar, and Cheng may not be prior art. Eylon was filed after the effective date of the present application. Kumar and Cheng have nothing to do with streaming software.

Eylon, Kumar, Schmeidler, and Cheng at least fail to teach or suggest any of the elements of the independent claims. Accordingly, the claims are allowable over any or all of the cited references.

In view of the foregoing remarks, Appellants submit that the pending claims are in condition for allowance and patentably define over the prior art, and urge the Board to overturn the Examiner's rejections.

Respectfully submitted,
Perkins Coie LLP



Date: November 20, 2006

William F. Ahmann
Registration No. 52,548

Correspondence Address:
Customer No. 22918
(650) 838-4300

CLAIMS APPENDIX

CLAIMS

1. (Previously Presented) A process for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment, comprising the steps of:

providing installation monitoring means for monitoring an installation process of said conventionally coded application program on a local computer system;

wherein said installation monitoring means gathers modification information including system registry modifications that said installation process makes to certain file paths in a system registry of said local computer system;

parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer; and

providing data set creation means for processing said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program.

2. (Original) The process of claim 1, wherein said data set creation means creates a runtime data set, said runtime data set consists of all regular application files and directories containing information about said regular application files.

3. (Original) The process of claim 2, wherein said data set creation means creates an initialization data set that is the first set of data streamed from said server to said client, said initialization data set prepares said client for streaming of said runtime data set.

4. (Original) The process of claim 2, wherein said directories contain lists of file names, file numbers, and the metadata associated with the files in a particular directory.

5. (Original) The process of claim 1, wherein said data set creation means creates a versioning table that contains a list of root file numbers and version numbers for tracking application patches and upgrades, and wherein each entry in said versioning table corresponds to one patch level of an application with a corresponding new root directory.

6. (Original) The process of claim 5, wherein said versioning table is sent to said client by said server, said client compares said versioning table with said client's root file number for the particular application program to find the necessary files required for a software upgrade or patch.

7. (Previously Presented) The process of claim 1, further comprising the step of: providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said modification information.

8. (Original) The process of claim 1, wherein said installation monitoring means monitors said application program as it runs and is being configured for a particular working environment on said local computer system and records common configurations of said application program thereby allowing said common configurations to be automatically duplicated on other client machines.

9. (Previously Presented) The process of claim 1, further comprising program profiling means for capturing a sequence of file blocks being accessed during normal execution of said application program.

10. (Original) The process of claim 9, wherein said sequence of file blocks is used to pre-cache frequently used blocks on said client before said application program is first used by a user.

11. (Original) The process of claim 9, wherein said sequence of file blocks is used to optimize large directories of files on said client for faster file accesses.

12. (Original) The process of claim 9, wherein said sequence of file blocks is tied to specific user input and wherein said client pre-fetches file blocks based on user input to said application program.

13. (Previously Presented) The process of claim 1, wherein said installation monitoring means records a state of said local computer system before said installation process begins to give a more accurate picture of any modifications that are observed by said installation monitoring means.

14. (Previously Presented) An apparatus for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment, comprising:

installation monitoring module that monitors the installation process of said conventionally coded application program on a local computer system;

wherein said installation monitoring module gathers modification information including system registry modifications that said installation process makes to certain file paths in the system registry of said local computer system;

a module that parameterizes said system registry modifications by replacing certain of said file paths with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer; and

data set creation module that processes said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program.

15. (Previously Presented) The apparatus of claim 14, wherein said data set creation module creates a runtime data set, said runtime data set consists of all

regular application files and directories containing information about said regular application files.

16. (Previously Presented) The apparatus of claim 15, wherein said data set creation module creates an initialization data set that is the first set of data streamed from said server to said client, said initialization data set prepares said client for streaming of said runtime data set.

17. (Original) The apparatus of claim 15, wherein said directories contain lists of file names, file numbers, and the metadata associated with the files in a particular directory.

18. (Previously Presented) The apparatus of claim 14, wherein said data set creation module creates a versioning table that contains a list of root file numbers and version numbers for tracking application patches and upgrades, and wherein each entry in said versioning table corresponds to one patch level of an application with a corresponding new root directory.

19. (Original) The apparatus of claim 18, wherein said versioning table is sent to said client by said server, said client compares said versioning table with said client's root file number for the particular application program to find the necessary files required for a software upgrade or patch.

20. (Previously Presented) The apparatus of claim 14, further comprising: a user interface that allows an operator to examine all changes made to said local

computer system during said installation process and to edit said modification information.

21. (Previously Presented) The apparatus of claim 14, wherein said installation monitoring module monitors said application program as it runs and is being configured for a particular working environment on said local computer system and records common configurations of said application program thereby allowing said common configurations to be automatically duplicated on other client machines.

22. (Previously Presented) The apparatus of claim 14, further comprising: program profiling module that captures sequence of file blocks being accessed during normal execution of said application program.

23. (Original) The apparatus of claim 22, wherein said sequence of file blocks is used to pre-cache frequently used blocks on said client before said application program is first used by a user.

24. (Original) The apparatus of claim 22, wherein said sequence of file blocks is used to optimize large directories of files on said client for faster file accesses.

25. (Original) The apparatus of claim 22, wherein said sequence of file blocks is tied to specific user input and wherein said client pre-fetches file blocks based on user input to said application program.

26. (Previously Presented) The apparatus of claim 14, wherein said installation monitoring module records a state of said local computer system before said

installation process begins to give a more accurate picture of any modifications that are observed by said installation monitoring.

27. (Previously Presented) A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for converting a conventionally coded computer application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment, comprising:

monitoring the installation process of said conventionally coded application program on a local computer system;

gathering modification information including system registry modifications that said installation process makes to certain file paths in the system registry of said local computer system;

parameterizing said system registry modifications by replacing certain of said file paths with parameters that are recognizable by said client; and

processing said modification information for converting said application program into a form for deceiving said client into allowing streaming of bits of said converted application program over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program.

28. (Previously Presented) The method of claim 27, further comprising creating a runtime data set, said runtime data set comprises regular application files and directories containing information about said regular application files.

29. (Previously Presented) The method of claim 28, further comprising creating an initialization data set that is the first set of data streamed from said server to said client, said initialization data set prepares said client for streaming of said runtime data set.

30. (Original) The method of claim 28, wherein said directories contain lists of file names, file numbers, and the metadata associated with the files in a particular directory.

31. (Previously Presented) The method of claim 27, wherein further comprising creating a versioning table that contains a list of root file numbers and version numbers for tracking application patches and upgrades, and wherein each entry in said versioning table corresponds to one patch level of an application with a corresponding new root directory.

32. (Original) The method of claim 31, wherein said versioning table is sent to said client by said server, said client compares said versioning table with said client's root file number for the particular application program to find the necessary files required for a software upgrade or patch.

33. (Previously Presented) The method of claim 27, further comprising providing a user interface that allows an operator to examine changes made to said local computer system during said installation process and to edit said system modification data and said file modification data.

34. (Previously Presented) The method of claim 27, further comprising monitoring said application program as it runs and is being configured for a particular working environment on said local computer system and records common configurations of said application program thereby allowing said common configurations to be automatically duplicated on other client machines.

35. (Previously Presented) The method of claim 27, further comprising capturing a sequence of file blocks being accessed during normal execution of said application program.

36. (Original) The method of claim 35, wherein said sequence of file blocks is used to pre-cache frequently used blocks on said client before said application program is first used by a user.

37. (Original) The method of claim 35, wherein said sequence of file blocks is used to optimize large directories of files on said client for faster file accesses.

38. (Original) The method of claim 35, wherein said sequence of file blocks is tied to specific user input and wherein said client pre-fetches file blocks based on user input to said application program.

39. (Previously Presented) The method of claim 27, wherein said installation monitoring means records a state of said local computer system before said installation process begins to give a more accurate picture of any modifications that are observed by said installation monitoring means.

40. (Previously Presented) A method for converting an application program into a data set suitable for streamed delivery across a network from a server to a client in a computer environment, the method comprising:

monitoring an installation process of an application program on a local computer system;

gathering modification information including information on modifications that said installation process makes to certain file paths in a system registry of said local computer system;

parameterizing said system registry modifications by replacing certain of said file paths in said system registry modifications with parameters that are recognizable by said client to re-direct requests for reading said system registry to a registry spoofer; and

processing said modification information for converting said application program into a data set suitable for deceiving said client into allowing streaming of bits of said data set over said network to said client such that said application program is capable of beginning execution on said client prior to downloading all of said application program.

41. (Previously Presented) The method of claim 40, further comprising creating a runtime data set, said runtime data set consists of regular application files and directories containing information about said regular application files.

42. (Previously Presented) The method of claim 41, further comprising creating an initialization data set that is the first set of data streamed from said server to said

client, said initialization data set prepares said client for streaming of said runtime data set.

43. (Previously Presented) The method of claim 41, wherein said directories contain lists of file names, file numbers, and the metadata associated with the files in a particular directory.

44. (Previously Presented) The method of claim 40, further comprising creating a versioning table that contains a list of root file numbers and version numbers for tracking application patches and upgrades, and wherein each entry in said versioning table corresponds to one patch level of an application with a corresponding new root directory.

45. (Previously Presented) The method of claim 44, wherein said versioning table is sent to said client by said server, said client compares said versioning table with said client's root file number for the particular application program to find the necessary files required for a software upgrade or patch.

46. (Previously Presented) The method of claim, 40, further comprising providing a user interface that allows an operator to examine all changes made to said local computer system during said installation process and to edit said system modification data and said file modification data.

47. (Previously Presented) The method of claim 40, further comprising monitoring said application program as it runs and is being configured for a particular working

environment on said local computer system and records common configurations of said application program thereby allowing said common configurations to be automatically duplicated on other client machines.

48. (Previously Presented) The method of claim 40, further comprising capturing a sequence of file blocks being accessed during normal execution of said application program.

49. (Previously Presented) The method of claim 48, wherein said sequence of file blocks is used to pre-cache frequently used blocks on said client before said application program is first used by a user.

50. (Previously Presented) The method of claim 48, wherein said sequence of file blocks is used to optimize large directories of files on said client for faster file accesses.

51. (Previously Presented) The method of claim 48, wherein said sequence of file blocks is tied to specific user input and wherein said client pre-fetches file blocks based on user input to said application program.

52. (Previously Presented) The method of claim 40, further comprising recording a state of said local computer system before said installation process begins.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None